

Algorithms and Data Structures

Jérémy Barbay

May 8, 2018

Contents

1	Introduccion	2
2	Desde la electronica a la programacion a los algoritmos	3
3	Introduccion a Teoria de la Computacion	3
3.1	CANC Programas Iterativos usando invariante	4
4	Algoritmos simples de ordenacion	5
5	Medidas de Complejidad	5
6	Disenio y Analisis de Algoritmos	6
7	Metodos Matematicos	6
7.0.1	Problema: Ecuacion de Stooage-sorting	7
8	Recursividad y Tabulacion	7
9	Dividir para reinar	7
10	Programacion Dinamica	8
10.1	CANC Algoritmos Avaros	8
11	Casos de estudio	9
12	Estructuras de datos elementales	10
12.1	Ejercicio	11
13	Listas, Pilas	12
14	Colas, colas de prioridad	12
15	arboles binarios, generales	13
16	Diccionarios: Busqueda secuencial/binaria, ABB	14
17	Arboles de Busqueda General y 2-3	15
18	Arboles 2-3, AVL, arboles digitales	16
19	B-Arboles	16
20	ABB optimo, Splay trees	16

21 Skip Lists	17
22 Arboles de busqueda digital	17
23 Bitmaps, hashing	17
24 Ordenamiento: cota inferior	18
25 Quickselect, heapsort	18
26 Radix sort	19
27 Grafos: Representacion, DFS y BFS	19
28 Distancias minimas (Dijkstra, Floyd, cerradura transitiva)	19
29 Arbol cobertor minimo (Kruskal, Prim)	19
30 Busqueda en texto: String Matching (Fuerza bruta, KMP, Boyer-Moore)	19

1 Introduccion

1. Quien soy

- Formacion en
 - Matematica y
 - Informatica
- Investigacion en
 - Teoria de la Computacion (Analisis de Algoritmos y Estructuras de Datos)
 - Pedagogia (<http://teachingislearning.cl>)
 - (un poco) hactivismo (Aaron Swartz day)

2. Quienes son

- Mayoria plan comun
- no todos futuros "computines"

3. El curso

- Objetivo: Aprender a formalizar problema, soluciones, medidas para comparar las soluciones (para la computacion, pero tambien para muchos otros temas, desde los procesos industriales hasta las tareas de la vida)
- Componente
 - teorico (clases magistrales, controles, examen) con
 - conneccion a la practica (5 tareas en java)
- Modalidades:
 - Evaluacion
 - * 5 tareas (primera [2018-03-16 Fri])
 - * 2 controles
 - * 1 examen final
 - Clases
 - * Charla interactivas los Martes y Jueves

- * Clases de ejercicios los Viernes
- Online
 - * publicacion de
 - apuntes "vivas" en Material Docente
 - plan de cursos (Corto) en Blog
 - * Uso intenso del foro:
 - los alumnos tienen que responder a las preguntas de los (otros) alumnos,
 - el rol del equipo docente es de verificar la validez de las respuestas.

2 Desde la electronica a la programacion a los algoritmos

1. Que hay en un computador
 - CPU
 - ALU
 - Memoria
 - Perifericos
2. Que hace un programa
 - Secuencia
 - Instrucciones
 - * Input (read)
 - * Output (write)
 - * Flow (if, then, else, while, until, . . .)
 - Librerias
3. Que hace un algoritmo?
 - trabajo preliminar antes de programar
 - trabajo en comun entre la programacion en multiples plataformas
 - tener un lenguaje sobre **los programas que no pueden existir**

3 Introduccion a Teoria de la Computacion

1. Arquitectura del Computador
 - (a) CPU
 - (b) Memoria
 - (c) Punteros
 - (d) Variables de Referencia
2. Herramientas: Estructuras de Datos (Concreto)
 - (a) Martillo (de metal)
 - (b) tornillador (cruciforme)
 - (c) Arreglo
 - (d) Punteros y Variables de Referencia
3. Aplicaciones: Tipos de Datos Abstractos (Abstraccion)

- (a) Martillar (concepto)
- (b) tornillar (concepto)
- (c) Conjunto
 - ☐ encuentra(clave)
 - ☐ tamano()
 - ☐ agrega(clave) (para conjunto dinamico)
 - ☐ borra(clave) (para conjunto dinamico)
 - ☐ encuentraMinimo() (para "cola de prioridad minima")
 - ☐ encuentraMaximo() (para "cola de prioridad maxima")
 - ☐ borraMinimo() (para "cola de prioridad minima")
 - ☐ borraMaximo() (para "cola de prioridad maxima")
- (d) Diccionario
 - ☐ encuentra(clave) -> dato
 - ☐ tamano()
 - ☐ agrega(clave,dato) (para diccionarios dinamicos)
 - ☐ borra(clave) (para diccionarios dinamicos)
 - ☐ encuentraProximo(clave) (para diccionarios ordenados)
 - ☐ encuentraPrevio(clave) (para diccionarios ordenados)

3.1 CANC Programas Iterativos usando invariante

<http://users.dcc.uchile.cl/~bebustos/apuntes/cc3001/Repaso/>

1. Introduccion

- (a) Que vimos la otra vez?
- (b) Apuntes en linea
 - <http://users.dcc.uchile.cl/~bebustos/apuntes/cc3001/>
 - <https://www.u-cursos.cl/ingenieria/2016/2/CC3001/2/enlaces/>

2. Syntaxis Java

- Variables
- Constantes
- Instrucciones
- Salida
- Condicionales
- Loops

3. Ejemplos

- Ordenar por insercion
- Ordenar por seleccion
- Ordenar por burbuja
- Calculo de x^n (mas sobre eso en recurrencias)

4. Conceptos de Programacion

- iterativo (Assembler, Basic)
- Orientada a Objetos (Java, python)
- Funcionales (ML, Caml, python)

4 Algoritmos simples de ordenacion

1. Problema de Ordenacion:
 - (a) input
 - (b) output
 - (c) aplicaciones
2. Algoritmos de Ordenacion
 - (a) Ordenar por insercion
 - (b) Ordenar por seleccion
 - (c) Ordenar por burbuja
3. Mas alla (OPCIONAL):
 - (a) Analisis por el Peor Caso
 - (b) Analisis por el Mejor Caso
 - (c) Analisis por el Caso Promedio

5 Medidas de Complejidad

1. "el" "Mejor" algoritmo
 - (a) El mejor algoritmo absoluto no existe
 - (b) Comparar de manera absoluta dos algoritmos es poco practico
2. Peor/Mejor/Promedio Caso (por n fijo)
 - (a) Mejor Caso
 - (b) Peor Caso
 - (c) Caso Promedio
3. Complejidades en el peor caso de Algoritmos de Ordenamiento Basicos
 - (a) Ordenar por insercion
 - (b) Ordenar por seleccion
 - (c) Ordenar por burbuja
4. Asimptoticas
 - (a) Complejidad en el peor caso por (tamano de input) n fijo define una funcion
 - (b) nos interesa el comportamiento por grandes valores de n
 - (c) **no** nos interesa mucho (en primera instancia) los factors constantes

6 Diseno y Analisis de Algoritmos

1. Recursividad
 - (a) x^n
 - (b) torre de Hanoi
 - (c) $x!$
 - (d) (OPT) Fibonacci
2. Backtracking
 - (a) Solucion de Laberinto
 - (b) minMax algorithm in game programming (Checkers)
 - (c) (OPT) SIMPLEX (optimizacion combinatoria)
3. (OPT) Problemas de las n reinas
 - (a) Recurrencia
 - (b) tiempo
 - (c) Espacio

7 Metodos Matematicos

1. Notaciones
 - O
 - Omega
 - Theta
 - o (OPCIONAL)
 - w (OPCIONAL)
2. Ecuaciones de Recursion
 - (a) (Algunas) Ecuaciones Non Lineales: Teorema Maestro: $T(n) = pT(n/q) + kn$
 - Desarrolla $T(n) = kn + pT(n/q)$
 - $T(n) = kn \sum_{i \in [0..j-1]} (p/q)^i + p^j T(1/q^j)$
 - Caso $p > q$
 - $T(n) \in O(n^{\log_q p})$
 - Caso $p = q$
 - $T(n) \in O(n \lg n)$
 - Caso $p < q$
 - $T(n) \in O(n)$
 - (b) Ecuaciones Lineales con coeficientes constantes: $f_n = A f_{n-1} + B f_{n-2} + C f_{n-3} + \dots$
 - Soluciones de la forma $f_n = \lambda^n$
 - para encontrar cual:
 - $f_n = \lambda^n$
 - Ejemplo: resuelve $f_n = f_{n-1} + f_{n-2}$
 - Equivale a resolver $\lambda^n = \lambda^{n-1} + \lambda^{n-2}$
 - divide por λ^{n-2}
 - obtiene el polinomio caracteristico

- le resuelve
- usa los casos
- (c) Ecuaciones de primer orden: $T(n) = aT(n-1) + b_n$
 - Caso $a = 1$: $T(n) = T(n-1) + b_n$
 - Telescopica:
 - $T(1) = T(0) + \sum_{i \in [1..n]} b_i$
 - Caso a general
 - $T(n) = a T(n-1) + b_n$
 - * divide by $1/a_n$
 - $T(n)/a_n = T(n-1)/a_{n-1} + b_n/a_n$
 - * Define $T'(n) = T(n) / a_n$
 - $T'(n) = T'(n-1) + b'_n$
 - (...)
 - $T(n) = a^n T(0) + \sum_{i=1}^n b_i a^{n-i}$
 - * Example Hanoi:
 - $T(n) = 2 T(n-1) + 1$
 - $T(n) = 2^n - 1$
- (d) Resolver otras Ecuaciones de recurrencia
 - Prueba por Induccion
 - Software: Matlab, Maple
 - Campo entero de Mathematica:
 - "Functional Analysis"
 - Fractals
 - etc...

7.0.1 Problema: Ecuacion de Stooage-sorting

8 Recursividad y Tabulacion

1. Fibonacci
2. Programacion Dinamica
3. Algoritmos Avaros

9 Dividir para reinar

1. Dividir para Reinar
 - (a) Basico: binary search
 - (b) General: Merge Sort
 - (c) Advanced: Optimal Boxes (Satellite imagery)
2. Notaciones Asimptoticas en Practica
 - (a) Inutiles para diferenciar "Insertion Sort" y "Selection Sort"
 - (b) Utiles para diferenciar "Merge Sort" and "Insertion Sort"
 - (c) Inutiles (de nuevo) para diferenciar "Merge Sort" y "Heap Sort"

10 Programacion Dinamica

1. Repaso:

(a) Recurrencias y Teorema Maestro

- Caso $p > q$: $T(n) \in O(n^{\log_a p})$
- Caso $p = q$: $T(n) \in O(n \lg n)$
- Caso $p < q$: $T(n) \in O(n)$

(b) Programas Recursivos

- Hanoi
- Fibonacci
- (Edit Distance)

2. Programacion Dinamica: resolver problemas de optimizacion (maximizacion o minimizaci3n de alguna funcion objetivo)

(a) Ejemplo: Multiplicacion de una secuencia de matrices

- A 100x10, B 10x100, C 100x10
- $(AB)C = 200.000$, $A(BC) = 20.000$

(b) Explosion de solucion ingenuas:

- los subproblemas se "traslapan" (overlapping problems)
- Tiempo en $4^n / n^{3/2}$
- Pero hay solamente $n(n-1) \in O(n^2)$ problemas distintos!

(c) memoization = Uso de memoria

- Espacio en $O(n^2)$
- Tiempo en $O(n^3)$

3. Ejemplos de Aplicaciones

(a) Multiplicacion de secuencia

(b) Longest Increasing Subsequence

(c) (Edit Distance)

10.1 CANC Algoritmos Avaros

CANC

1. Algoritmos Avaros ("Greedy Algorithms")

- para resolver problemas de optimisacion
- busca optimum **local**, simple de programar:
 - toma decisiones en base a informacion local
 - nunca cambia una decision pasada
- no siempre optimal (e.g. cuando algunos optimos locales no son globales)
- Ejemplo:
 - (a) camino mas corto
 - (b) assignacion de actividades (ejemplo de las apuntes)

2. Estudio de Caso: Subsecuencia de Suma Maxima

- Definicion:
 - Dados enteros A_1, \dots, A_n

- Encontrar i, j tal que $\sum_{k \in [i..j]} A_k$ es maximo
- Ejemplo
 - $S = -2, 11, -4, 13, -5, -2$
 - Respuesta: 20

3. Soluciones

- (a) Fuerza Bruta: $O(n^2)$
- (b) Fuerza Bruta mejorado: $O(n^2)$
- (c) Dividiendo el problema: $O(n \lg n)$
 - $T(n) = 2T(n/2) + O(n)$
- (d) Algoritmo Eficiente: $O(n)$

11 Casos de estudio

1. Subsecuencia de Suma Maxima

- Importancia del orden (arreglo en los slides de Benjamin)

2. Multiplicacion de dos Matrices

- OJO: problema distinto de la optimizacion del calculo del producto de una cadena de matrices
- Simple algoritmo: $O(n^3)$
- Se puede mejorar?
 - no mejor que $O(n^2)$ -> la complejidad del problema es a dentro de $\Omega(n^2)$
 - problema abierto por mucho tiempo de mejorar $O(n^3)$ o $\Omega(n^2)$
 - en 1960, Strassen mostro como mejorar $O(n^3)$ por dividir y conquistar
- $T(N) \in 7T(N/2) + O(N^2)$ -> $T(N) \in O(N^{\log_2(7)}) \subseteq O(N^{2.81})$
- Nota:
 - Todavia no es abierto el problema de la complejidad
 - Algoritmo de Strassen mejor solamente cuando N es muy grande
 - El algoritmo es numericamente inestable.
 - La multiplicacion de 2 matrices tiene muchas aplicaciones sorprendantes (Transforma de Fourier)
 - The (simplified) Teorema maestro as previously taught applies to recurrences of the form $T(n) = pT(n/q) + kn$
 - Strassen algorithms yields a recurrence in $T(N) \in 7T(N/2) + O(N^2)$
 - The result of $O(N^{2.81})$ is still valid by the more general master theorem (e.g. as described on https://en.wikipedia.org/wiki/Master_theorem), for equations of the form $T(n) = pT(n/q) + f(n)$ where $f(n) \in O(n^c)$ where $c < \log_p q$.

3. LCSS = "Longest Common Sub Sequence" = "Subsecuencia comun mas larga"

- (a) Contexto:
 - i. Aplicaciones:
 - Comparar dos secuencias de ADN o ARN
 - Comparar dos tareas submitidas por alumnos
 - ii. En general, es una medida (entre otras) para determinar si dos secuencias son similares:
 - si una es una subsecuencia de la otra

- costo de transformar una en otra (distancia de edicion o "Edit Distance", cf tarea 3)
- encontrar una tercera que se parezca a ambas

iii. Definicion:

Subsecuencia la secuencia con cero o mas elementos dejados fuera

Formalmente:

- Z es subsecuencia de X si existe secuencia de indices creciente de X tal que $(i_1, \dots, i_{|Z|}), \forall j \in [1..k] z_j = x_{i_j}$

Subsecuencia comun Z es subsecuencia comun de X e Y si es subsecuencia de X y de Y .

el problema es de encontrar la subsecuencia comun mas grande entre dos secuencias X y Y (de tamaños n y m)

(b) Algoritmos: Cual algoritmo pueden imaginar?

i. Fuerza Bruta

- Cuantas subsecuencias tiene una secuencia de n elementos?
 - en el peor caso (e.g. sin repeticiones de simbolos)

ii. Dividir (por conquistar)

- Define $X_i = (x_1, \dots, x_i)$
- Subproblemas: encontrar la subsecuencia mas larga de subfijos

Theorema – Sea X_m e Y_n secuencias, Z_k una LCS de X e Y

- * Si $x_m = y_n$,
 - $z_k = x_m = y_n$ y Z_{k-1} es una LCS de X_{m-1} e Y_{n-1}
 - (acordense que los elementos de Z no tienen que ser consecutivos en X o Y)
- * Si $x_m \neq y_n$,
 - $z_k \neq x_m$ implica que X es una LCS de X_{m-1} e Y_n .
 - $z_k \neq y_n$ implica que X es una LCS de X_m e Y_{n-1} .
- El Teorema sugiere una solucion recursiva de grado 2 (dos llamadas recursivas al maximo)
- Matriz C de $m \times n$ entradas, definidas por
 - $c[i, j] =$
 - * 0 si $i = 0$ y $j = 0$
 - * $c[i - 1, j - 1] + 1$ si $i, j > 0$ y $x_i = y_j$
 - * $\max\{c[i, j - 1], c[i - 1, j]\}$ si $i, j > 0$ y $x_i \neq y_j$
- Rendimiento
 - tiempo en $O(nm)$
 - espacio en $O(n)$

4. Eso es la fin de la parte sobre paradigmos de programacion

- Sigamos en la proxima session con Estructuras de Datos!

12 Estructuras de datos elementales

1. Conjunto(s)

(a) Conjunto desordenado dinamico

- encuentra(clave)
- tamano()
- agrega(clave)
- borra(clave)

(b) Cola de prioridad minima (dinamica)

- encuentra(clave)
- tamano()
- agrega(clave)
- borra(clave)
- encuentraMinimo()
- borraMinimo()

2. Diccionario(s)

(a) Diccionario statico ordenado

- encuentra(clave) -> dato
- tamano()
- encuentraProximo(clave)
- encuentraPrevio(clave)

(b) Diccionario dinamico ordenado

- encuentra(clave) -> dato
- tamano()
- agrega(clave,dato)
- borra(clave)
- encuentraProximo(clave)
- encuentraPrevio(clave)

12.1 Ejercicio

Cuales son lo Tipos de Datos Abstractos correspondiendo a estas Estructuras de Datos? (i.e. Cuales problemas pueden resolver estas soluciones?)

1. Arreglo desordenado (mas una variable para el tamano)

- agregar
- tamano
- buscar (dificil)
- borrar (una vez encontrado)
- proximoMasGrande(clave) dificil
- encuentra minimo(clave) dificil

2. Arreglo ordenado

3. Lista Enlazada

4. Arbol Binario (de Busqueda)

5. Arbol General (de Busqueda)

13 Listas, Pilas

1. Listas (Encadenadas)

- Busqueda
- Insercion
- Deleccion

2. Pilas

- ADT o DS?
- Implementaciones
 - Listas
 - Arreglo
- Aplicaciones
 - pasaje de parametros a funciones

3. Fila

- LIFO, FILO, LILO, FIFO: cuantas posibilidades?

14 Colas, colas de prioridad

1. Conceptos

- (a) Niveles de Abstraccion
- (b) Typos de Datos Abstractos ("Abstract Data Type" ADT)
- (c) Data Structure

2. Aplicaciones

(a) Fila (File)

- ADT: FIFO = First In First Out
- DS:
 - en lista
 - en arreglo
- Rendimientos

(b) Pilas (stack)

- ADT: LIFO = Last In First Out
- DS:
 - en lista
 - en arreglo
- Rendimientos

(c) Other ADTs:

- FILO = First In Last Out?
- LILO = Last In Last Out?
- Otros?

3. Listas

(a) Descripcion

(b) ADT o DS?

(c) Variantes

- "Double Linked List"
- Lista de Listas
- Skiplists

4. Colas

(a) Tipo de Datos Abstractos (TDAs)

- Cola
- Cola de prioridad

(b) Estructuras de Datos

- Lista
- Pila
- Cola
- Cola de prioridad

5. Cola de Prioridad mas en detalles

- CorrectUp
- CorrectDown
- Heapify

15 arboles binarios, generales

1. Definiciones

(a) Arboles Ordinales

- Nodos Internos y Externos (Hojas)
- Altura
- Profundidad de un nodo o de una hoja

(b) Arboles Cardinales (y Binarios en particular)

- general, o
- Balanceado, o

- Completo y Quasi completo
- (c) Aplicaciones
- Expresiones de calculo
 - notacion polacka invertida
 - Expresiones con parenthesis
 - Heaps
 - Diccionarios
 - Arboles De Busqueda
 - Red-Black Arboles,
 - (2-3)-Arboles, (k-(2k-1))-Arboles
 - B-Arboles
 - ...
2. Propriedades
- (a) Cantidad de nodos internos vs cantidad de nodos externos en un arbol binario?
- $e = 1 + i$
- (b) Altura de un arbol completo con n nodos, si
- binario? $h = \log_2(n + 1) - 1 \in O(\log_2 n)$
 - (k-ary)? (homework)
 - quasi-completo binario? $h = \lceil \log_2(n + 1) \rceil - 1 \in O(\log_2 n)$
 - balanceado?
3. Nociones Utiles
- Pre-orden
 - In-Orden (binario)
 - Post-Orden
 - DFUDS

16 Diccionarios: Busqueda secuencial/binaria, ABB

1. TDA(s) Diccionario
- (a) TDA Diccionario Estatico
- `compile(conjunto de pares)`
 - `find(key)`
- (b) TDA Diccionario Dinamico
- `isEmpty()`
 - `add(key,data)`
 - `remove(key,data)`
 - `find(key)`

(c) TDA Diccionario (Dinamico) Ordenado

- ☒ isEmpty()
- ☒ add(key,data)
- ☒ remove(key)
- ☒ find(key)
- ☒ findNext(key)
- ☐ rank(key)
- ☐ select(rank)

2. Tecnicas para Diccionario

(a) Arreglo desordenado

- Busqueda Secuencial
- moveToFront

(b) Arreglo ordenado

- Busqueda Secuencial
- Busqueda Binaria
- Busqueda Doblada
- Otras Busquedas

(c) Arbol Binario de Busqueda

- Busqueda en arbol de busqueda

17 Arboles de Busqueda General y 2-3

1. TDA Diccionario (Dinamico) Ordenado

- ☒ isEmpty()
- ☐ add(key,data)
- ☐ remove(key)
- ☒ find(key)
- ☒ findNext(key)

2. Estructuras de Datos para Diccionario :AVL ((Georgy Adelson-Velsky and Evgenii Landis' tree, 1962)

- ☐ isEmpty()
- ☐ add(key,data)
- ☐ remove(key)
- ☐ find(key)
- ☐ findNext(key)

3. (2,3)-Arboles

- (a) ☐ find(key) $3h \in O(h)$
- (b) ☐ findNext(key) $O(h)$
- (c) ☐ add(key,data) $O(h)$
- (d) ☐ remove(key) $O(h)$

4. Combinatoria

18 Árboles 2-3, AVL, árboles digitales

1. (d,2d-1)-Árboles

- (a) ☐ find(key)
- (b) ☐ add(key,data)
- (c) ☐ remove(key)
- (d) ☐ findNext(key)

2. Finger Search Trees

- (a) ☐ find(key)
- (b) ☐ add(key,data)
- (c) ☐ remove(key)
- (d) ☐ findNext(key)

19 B-Árboles

20 ABB optimo, Splay trees

1. ☒ "Move To Front" en Arreglos Ordenados

- (a) Definición de distribución
- (b) Promedio sobre input
- (c) ventajas sobre análisis en el peor caso

2. ☐ Árboles de Búsqueda Binarios Optimos (ABB optimos)

- (a) Definición
- (b) Computación (Programación Dinámica!)
- (c) Análisis: $O(n^3)$, $O(n^2)$

3. ☐ Splay Trees (AVLs que cambian también cuando se busca)

- (a) Motivación
- (b) Definición
- (c) Análisis: logros y problemas abiertos

21 Skip Lists

1. Algoritmos y Estructuras de Datos aleatorizados
 - (a) Algoritmos y Estructuras de Datos determinísticos
 - (b) Instrucciones aleatorizadas
 - (c) Analisis: Peor caso vs Promedio
 - i. ☐ sobre instancias
 - ii. ☒ sobre aleatorizacion
2. SkipLists: diseño
 - (a) ☒ Listas enlazadas
 - (b) ☒ Resumen exacto de Listas enlazadas
 - (c) ☒ Resumen aproximado de listas enlazadas
3. Skiplists: analisis
 - (a) tiempo de busqueda
 - (b) tiempo de inserción
 - (c) tiempo de delección

22 Arboles de busqueda digital

23 Bitmaps, hashing

1. Valores y Comparaciones
 - (a) Algoritmos en el modelo de comparaciones (e.g. busqueda binaria)
 - (b) Algoritmos afuera del modelo de comparaciones (e.g. busqueda por interpolacion)
 - (c) Frecuencia de colisiones: Paradojo de los cumpleaños
2. Tablas de Hash
 - (a) ☐ Encadenamiento
 - Listas enlazadas en cada cedula
 - Hashing con listas mezcladas
 - (b) ☐ Direccionamiento abierto ("Open directing" but "closed table")
 - Linear Probing
 - Quadratic Probing
 - Hashing con doble funcion de hash
3. Detalles tecnicos

- (a) Borrar
- (b) Funciones de Hash
 - Suma de caracteres
 - funcion de hash aleatorizada $h_{a,b}(k) = ak + b \bmod p \bmod N$
- (c) Analisis amortizada

24 Ordenamiento: cota inferior

1. Busqueda Desordenada
2. Busqueda Ordenada
3. Ordenamiento

25 Quickselect, heapsort

1. Heap Sort
 - (a) Review Priority Queues and Dictionaries
 - (b) Using Priority Queues for Sorting (Heapify)
 - (c) Using Dictionaries for Sorting
2. Quick Sort
 - (a) Partitioning an array by a pivot
 - linear time median
 - (b) Divide and Conquer Sorting no 2
 - (c) Detecting very frequent elements
3. Quick Select
 - (a) Definitions
 - Rank
 - Select
 - (b) Select Algorithms
 - $O(n \lg n)$
 - $O(n)$
 - (c) Lazy Data Structures for Rank and Select Queries (Online)

26 Radix sort

1. Counting Sort

- Value Based Sorting algorithms: (relatively) small domain
- Complexity in function of n and σ

2. Hash Sort?

- large domain but Small effective domain

3. Radix Sort

27 Grafos: Representacion, DFS y BFS

- Grafos
 - ADT
 - DS
 - * Matrix
 - * Listas
 - * Otras
 - Aplicaciones

28 Distancias minimas (Dijkstra, Floyd, cerradura transitiva)

- https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
- https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm

29 Arbol cobertor minimo (Kruskal, Prim)

- https://en.wikipedia.org/wiki/Kruskal%27s_algorithm
- https://en.wikipedia.org/wiki/Prim%27s_algorithm

30 Busqueda en texto: String Matching (Fuerza bruta, KMP, Boyer-Moore)

1. Pattern Matching: Definition and Brute Force Algorithm

- Definition
- Brute Force
- Improvements
 - Indexing the Pattern (explored in this lecture)
 - Indexing the text (current research)

- Indexing both

2. Knuth-Morris-Pratt (KMP) Algorithm

- Knuth:
 - Father of Theoretical Computer Science
 - Original Programmer of \TeX
 - Author of "The Art of Computer Programming", a reference in the field
- Ideas:
 - Failure function
- Complexity:
 - $O(n+m)$ in worst and best case.

3. Boyer-Moore-Horspool (BMH) Algorithm

- Ideas:
 - Right to Left
 - Simplified Failure function
- Complexity
 - $O(nm)$ in worst case
 - $O(n/m)$ in best case and "on average"

4. Beyond the course

(a) Combination of KMP with BMH: BMS

- $O(n+m)$ in worst case
- $O(n/m)$ in best case and "on average"
- $\Omega(n/m)$ lower bound anyway

(b) Automata (Extra)

- Completely indexing the pattern
- $O(n+m^2)$ worst case

(c) Pattern Matching with Partial MisMatch